DW_fp_div

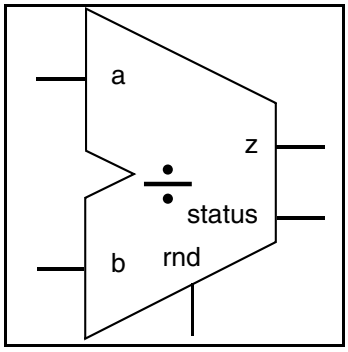
Floating-Point Divider

Version, STAR, and myDesignWare Subscriptions: IP Directory

Features and Benefits

Revision History

- The precision format is parameterizable for either IEEE single, double precision, or a user-defined custom format
- Hardware for denormal numbers of IEEE 754 standard is provided
- Configurable to be fully compliant with the IEEE Std 754-1985 standard
- Configurable for NaN representation compatible with the IEEE Std 754-2008 standard (controlled by the *ieee_compliance* parameter)
- Faithful rounding with 1 ulp error is supported by parameter
- DesignWare datapath generator is employed for better timing and area



Description

DW_fp_div is a floating-point divider that divides two floating-point operands: `a` by `b` to produce a floating-point quotient, `z`.

Component pins are described in Table 1-1 and configuration parameters are described in Table 1-2.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
a	*exp_width* + *sig_width* + 1 bits	Input	Dividend
b	*exp_width* + *sig_width* + 1 bits	Input	Divisor
rnd	3 bits	Input	Rounding mode; supports all rounding modes described in the *Datapath Floating-Point Overview*; The `rnd` port takes effect only when *faithful_round* = 0.
z	*exp_width* + *sig_width* + 1 bits	Output	Quotient of A/B

Table 1-1 Pin Description (Continued)

Pin Name	Width	Direction	Function
status	8 bits	Output	■ Status flags corresponding to `z`; for details, see STATUS Flags in the *Datapath Floating-Point Overview* ■ status[6]: Underflow before rounding (UBR) status flag When enabled by the *en_ubr_flag* parameter, this flag indicates when the absolute value of a non-zero result, computed as though both the exponent range and the precision of the significand were unbounded, would lie strictly between ±MinNorm (minimum normalized value representable in the FP format defined by *sig_width* and *exp_width*). ■ status[7]: Indicates divide-by-zero operation

Table 1-2 Parameter Description

Parameter	Values	Description
sig_width	2 to 253 bits Default: 23	Word length of fraction field of floating-point numbers `a`, `b`, and `z`
exp_width	3 to 31 bits Default: 8	Word length of biased exponent of floating-point numbers `a`, `b`, and `z`
ieee_compliance	0, 1, or 3 Default: 0	Level of support for the IEEE Std 754 standards: ■ 0: No support for NaNs and denormals; NaNs are considered infinities and denormals are considered zeros ■ 1: Fully compliant with the IEEE Std 754-1985 standard, including support for NaNs and denormals ■ 2: Reserved ■ 3: Fully compliant with the IEEE Std 754-1985 standard plus NaN representation that matches the IEEE Std 754-2008 standard[a] For details, see Compatibility with IEEE Std 754 Standards in the *Datapath Floating-Point Overview*
faithful_round	0 or 1 Default: 0	Choose either a specific rounding mode (set by `rnd`) or a general rounding mode that allows maximum 1 ulp error ■ 0: Rounding mode is specific, as set by the `rnd` port; this choice increases the size of the resulting implementation. ■ 1: Rounding mode is general and, for *sig_width* ≤ 23, allows a maximum of 1 ulp error[b]; this choice decreases the size of the resulting implementation. When *faithful_round* = 1, note the following: - The inexact status flag in the output is not meaningful. - The UBR status flag (`status`[6]) is always 0 (disabled). - The other status flags will match one of the possible outputs for the calculation when *faithful_round* = 0.

Table 1-2 Parameter Description (Continued)

Parameter	Values	Description
en_ubr_flag	0 or 1 Default: 0	Controls when the flag in `status`[6] is enabled ■ 0: The UBR status flag `status`[6] is disabled (the value of this flag is always 0). ■ 1: When this parameter is 1 and *faithful_round* = 0, `status`[6] carries the value of the UBR flag. If *faithful_round* = 1, even when *en_ubr_flag* = 1, the UBR flag in `status`[6] bit is always 0.

a. Propagating payload information to the output during the NaN process, which is an optional feature specified in the IEEE Std 754-2008 standard, is not supported.
b. When *faithful_round* = 1 and *sig_width* > 23, the result can exceed 1 ulp for some corner cases. Configurations tested for this parameter range do not show errors larger than 2 ulps.

Table 1-3 Synthesis Implementations[a]

Implementation Name	Function	License Feature Required
rtl	Synthesis model (Digit-recurrence Method)	DesignWare
str	Synthesis model (Newton-Raphson Method) NOTE: This implementation is disabled when *sig_width* ≤ 10. In this scenario, use the rtl implementation because it provides better QoR.	DesignWare

a. During synthesis, Design Compiler selects the appropriate architecture for your constraints. However, you may force Design Compiler to use one of the architectures described in this table.

Table 1-4 Simulation Models

Model	Function
DW02.DW_FP_DIV_CFG_SIM	Design unit name for VHDL simulation
dw/dw02/src/DW_fp_div_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW_fp_div.v	Verilog simulation model source code

Suppressing Warning Messages During Verilog Simulation

The Verilog simulation model includes macros that allow you to suppress warning messages during simulation.

To suppress all warning messages for all DWBB components, define the DW_SUPPRESS_WARN macro in either of the following ways:

- Specify the Verilog preprocessing macro in Verilog code:

  ```
  `define DW_SUPPRESS_WARN
  ```

- Or, include a command line option to the simulator, such as:

  `+define+DW_SUPPRESS_WARN` (which is used for the Synopsys VCS simulator)

The warning messages for this model include the following:

- If an invalid rounding mode has been detected on `rnd`, the following message is displayed:

  ```
  WARNING: <instance_path>:
      at time = <timestamp>: Illegal rounding mode.
  ```

  To suppress this message, use the DW_SUPPRESS_WARN macro explained earlier.

Related Topics

- Datapath Floating-Point Overview
- DesignWare Building Block IP User Guide

HDL Usage Through Component Instantiation - VHDL

```
library IEEE,DWARE;
use IEEE.std_logic_1164.all;
use DWARE.DW_Foundation_comp.all;
-- If using numeric types from std_logic_arith package,
-- comment the preceding line and uncomment the following line:
-- use DWARE.DW_Foundation_comp_arith.all;

entity DW_fp_div_inst is

  generic (
    inst_sig_width       : POSITIVE := 23;
    inst_exp_width       : POSITIVE := 8;
    inst_ieee_compliance : INTEGER := 0;
    inst_faithful_round  : INTEGER := 0;
    inst_en_ubr_flag : INTEGER := 0
  );
  port (
    inst_a      : in std_logic_vector(inst_sig_width+inst_exp_width downto 0);
    inst_b      : in std_logic_vector(inst_sig_width+inst_exp_width downto 0);
    inst_rnd    : in std_logic_vector(2 downto 0);
    z_inst      : out std_logic_vector(inst_sig_width+inst_exp_width downto 0);
    status_inst : out std_logic_vector(7 downto 0)
  );

end DW_fp_div_inst;


architecture inst of DW_fp_div_inst is

begin

  -- Instance of DW_fp_div
  U1 : DW_fp_div
  generic map (
    sig_width => inst_sig_width,
    exp_width => inst_exp_width,
    ieee_compliance => inst_ieee_compliance,
    faithful_round => inst_faithful_round,
    en_ubr_flag => inst_en_ubr_flag
  )
  port map (
    a => inst_a,
    b => inst_b,
    rnd => inst_rnd,
    z => z_inst,
    status => status_inst
  );
```

```
end inst;


-- pragma translate_off
configuration DW_fp_div_inst_cfg_inst of DW_fp_div_inst is
   for inst
   end for;
end DW_fp_div_inst_cfg_inst;
-- pragma translate_on
```

HDL Usage Through Component Instantiation - Verilog

```
module DW_fp_div_inst( inst_a, inst_b, inst_rnd, z_inst, status_inst );

parameter sig_width = 23;
parameter exp_width = 8;
parameter ieee_compliance = 0;
parameter faithful_round = 0;
parameter en_ubr_flag = 0;


input [sig_width+exp_width : 0] inst_a;
input [sig_width+exp_width : 0] inst_b;
input [2 : 0] inst_rnd;
output [sig_width+exp_width : 0] z_inst;
output [7 : 0] status_inst;

  // Instance of DW_fp_div
DW_fp_div #(sig_width, exp_width, ieee_compliance, faithful_round, en_ubr_flag) U1
( .a(inst_a), .b(inst_b), .rnd(inst_rnd), .z(z_inst), .status(status_inst)
);

endmodule
```

Revision History

For notes about this release, see the *DesignWare Building Block IP Release Notes*.

For lists of both known and fixed issues for this component, refer to the STAR report.

For a version of this datasheet with visible change bars, click here.

Date	Release	Updates
April 2022	DWBB_202203.1	■ Added the description of `status`[6] to Table 1-1 on page 1 ■ Added the *en_ubr_flag* parameter to Table 1-2 on page 2 and to the examples on page 5 and page 7
January 2022	DWBB_202106.5	■ Adjusted description of the str implementation in Table 1-3 on page 3
October 2020	DWBB_202009.1	■ For enhanced NaN compatibility with the IEEE Std 754 standards, added a new value for *ieee_compliance* in Table 1-2 on page 2
July 2020	DWBB_201912.5	■ Adjusted the description of the *ieee_compliance* parameter in Table 1-2 on page 2 ■ Added "Suppressing Warning Messages During Verilog Simulation" on page 4 ■ Added the *faithful_round* parameter to the examples on page 5 and page 7
April 2020	DWBB_201912.3	■ Updated the value range for the `sig_width` parameter in Table 1-2 on page 2 ■ Updated the description of `rnd` in Table 1-1 on page 1 and *faithful_round* in Table 1-2 on page 2 ■ For STAR 3124396, added a footnote to Table 1-2 on page 2 to update the error range when *faithful_round* = 1 and *sig_width* > 23. This update is based on a limitation found during the investigation of STAR 3124396. Also, Table 1-5, "Error Ranges ($\varepsilon = z - a/b$)" was removed.
July 2017	DWBB_201612.5	■ For STAR 9001189734, added Note to description of str implementation in Table 1-3 on page 3 ■ Added this Revision History table

Copyright Notice and Proprietary Information

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at https://www.synopsys.com/company/legal/trademarks-brands.html.

All other product or company names may be trademarks of their respective owners.

Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.
www.synopsys.com